

# InnoDB Architecture and Performance Optimization

Peter Zaitsev, Percona Ltd



**OPEN SOURCE DATABASE  
CONFERENCE 2006**

Learn • Share • Network • Benefit

# About Speaker

- Peter Zaitsev
  - Co-Founder Percona Ltd
    - Company specializing in MySQL and LAMP Performance Optimization
  - Spent 4.5 years with MySQL AB
    - High Performance Group Manager



# About Presentation

- What is Innodb
  - This is not MySQL Conference so I can't assume you know it :)
  - Innodb Architecture Basics
  - Innodb Performance Tuning



# About Audience

- How many of you
  - Using MySQL ?
  - Using Innodb ?
  - What is your primary storage engine ?
  - Have ever moved some of data from MyISAM to Innodb ?



# My InnoDB Background

- Using in InnoDB before it was integrated to MySQL main tree
  - Found many bugs back then
- Managing over 1TB worth of InnoDB data back in 2001
- InnoDB consulting, benchmarking activities, performance patches, talks while in MySQL





# About InnoDB



**OPEN SOURCE DATABASE  
CONFERENCE 2006**

Learn • Share • Network • Benefit

# MySQL Server Architecture

- MySQL Server can be viewed as two layers
  - SQL Layer
    - Query parsing, optimization, execution
  - Storage Engine
    - Data storage, locking, transactional management, recovery
- You can compare Storage Engine to file System in operation system
  - Just more complicated



# InnoDB Features

- ACID Transactional Storage engine for MySQL
- Multi Versioning
- Foreign Keys
- Row level locking
- Automatic Crash recovery
- Optimized for high performance
- Stable and secure



# General Purpose Storage Engine

- InnoDB is best general purpose storage engine available at this point
  - Number of contenders are in development (Solid, Falcon and more)
- Should have been default storage engine for technical reasons
  - Not owned by MySQL So that did not happen



# Why InnoDB

- It is not Only about transactions
  - No partially executed statements
  - Automatic fast crash recovery
    - Repair table on 20GB MyISAM Table takes a while
  - Row level locks and Multi versioning
    - No problem of “can I run this query now?”
  - Data clustering by Primary key
  - Sometimes better performance



# InnoDB Alternatives Future

- MySQL Develops Alternatives for InnoDB
- Will takes years to reach same level of stability and need to beat performance
  - Storage engine bugs may mean losing your data not just crashes
- Hard to have 100% same behaviour
  - Migration pains
- InnoDB is GPL - owner does not matter





# InnoDB Architecture



**OPEN SOURCE DATABASE  
CONFERENCE 2006**

Learn • Share • Network • Benefit

# On disk layout

- Single tablespace of multiple concatenated files
  - Or tablespace per table with special option
- Fixed size log files (2 or more files)
  - User in circular fashion
  - “physiological” logging - space efficient
- .frm file for each table - MySQL level info



# Tablespace format

- 16KB Fixed page size for data and indexes
- Tablespace contains multiple “segments”
  - Each table has one segment for data two for each index (leaf and non-leaf nodes).
  - Segment space allocated page by page up to 32 pages and in 64 page “extent” afterwards
- Other structures: Undo segments, DoubleWrite buffer area



# Log Format

- Log File contains dynamic length records
  - Not pages
- Each record identifies page and operation to perform on that page.
- Requires pages to be in consistent state
- Compromise between physical and logical
  - So called “physiological”



# Table Format

- Table data is stored in BTREE
- Clustered by Primary Key
- If no PK defined, internal PK allocated
  - 6 bytes in length.
- Up to 8K rows stored in the data pages
  - Larger rows partially stored in clustered index leaf segment
    - Each long column in its own set of pages
  - Prefix for all blobs stored on the page



# Index Format

- Row is referenced from Index by Primary Key
  - And sorted by Primary key as last index part
  - So short primary keys are very important
- Indexes are not packed
  - So can be much larger compared to MyISAM
- All unpurged rows versions kept in index
  - Many versions slow down index scan



# Multi Versioning

- New row versions replace old ones on the same page, same physical location
  - Avoid fragmentation with same row updates
- Old row versions are kept in Undo segment
  - Automatically purged when no one needs them
- Undo segment grows automatically
- Cached in buffer pool just as other pages



# Recovery

- Fuzzy Checkpointing
  - Avoid performance gaps during checkpoints
- Redo unflushed actions by log files
- Undo - Roll back uncommitted transactions
  - Use undo logs in system tablespace to restore appropriate versions
  - Can take a while if transaction was not



# Isolation Modes

- Support for wide range of isolation modes
  - From READ-UNCOMMITTED to SERIALIZABLE
- No phantom rows in REPEATABLE-READ
- Locking read modifiers for SELECT
  - SELECT FOR UPDATE
  - SELECT LOCK IN SHARE MODE
  - By default selects not locking



# Locking

- Row level locking with no escalation
- Non-locking selects by default
  - Less needs for locks
- Very efficient bitmap based locks
- Automatic instant deadlock detection
- Variable to control maximum lock wait time as an extra



# Locking mode

- Only Pessimistic locking support
  - Lock rows as they need to be locked
    - Wait if they are locked by other processes
- Optimistic Locking works as
  - Assume no lock conflicts will happen
  - Check if it is the case in the end
    - Rollback if conflict is discovered
- Pessimistic - better for high concurrency



# Memory Usage

- Buffer Pool
  - Cache for data and index pages, Lock structures, adaptive hash indexes
- Additional Pool
  - Dictionary, various meta data
- Log Buffer
  - Storage of log records before they are flushed to the disk



# Asynchronous operation

- Most work done from MySQL thread executing query
- Main Thread - Does purging, buffer flushes, log flushes, insert buffer merges
- Read Ahead Thread
  - Doing read ahead
- Deadlock detection and monitoring threads



# Latching and Synchronization

- Implements highly efficient mutexes
  - Better performance, more control
- Performs thread throttling for high concurrency loads
  - Limits about of threads in InnoDB kernel
  - Remaining threads are queued
  - Can be disabled if needed



# Special Features

- Adaptive Hash indexes
  - Ultra fast lookups for data in buffer pool
- Insert Buffer
  - Faster inserts in secondary indexes
- DoubleWrite Buffer
  - Extra data protection from partial page writes





# Innodb Performance Optimization



**OPEN SOURCE DATABASE  
CONFERENCE 2006**

Learn • Share • Network • Benefit

# Do not use defaults

- Default settings are for toy databases
  - 8MB buffer pool, 10MB logs size
  - Not enough for any serious load
- InnoDB is affected by buffer sizes much more than MyISAM
  - Advanced caching
  - Synchronous IO



# Sizing Buffers

- **innodb\_buffer\_pool\_size**
  - Typical value 60-80% of memory
    - If InnoDB is only your storage engine
- **key\_buffer\_size**
  - May be still needed for temporary tables
  - Some 32MB is enough
- **log\_buffer\_size**
  - 4-8MB is enough for most cases



# Sizing Log Files

- Larger log files - better performance
  - Reduces amount of flushes needed
  - Increases recovery time
- Use log files which give you recovery time you need
  - Combined size about 1GB is reasonable value
- Resizing is a bit complicated
  - Removing old log files so new ones are created



# Optimizing IO

- Avoid Double Buffering
  - Same data cached in OS cache and buffer pool
    - Waste of memory
  - InnoDB cache is much more efficient
  - Use unbuffered IO
- Linux
  - `innodb_flush_method=O_DIRECT`
  - May slow things down write performance



# Optimizing Commits

- ACID Commits require flush to the disk
  - Expensive, limited to 100-250/sec uncached
  - RAID with battery backup cache can improve dramatically (1000/sec+)
- Group commit broken in MySQL 5.0
  - Unless you disable binary log and sacrifice recovery from backup
- **`innodb_flush_log_at_trx_commit=2`**



# Other Variables

- **innodb\_additional\_mem\_pool\_size**
  - Used for data dictionary and other data
  - Automatically increased as needed
  - Do not set too high, avoid memory waste
- **innodb\_thread\_concurrency**
  - Limit number of Threads running in kernel
  - $2 * (\text{NumCPUs} + \text{NumDisks})$  - in theory
  - Optimal may be much smaller in practice



# innodb\_file\_per\_table

- Use its own tablespace for each table
- System tablespace is still used for undo segments and metadata
- Easier to backup, reclaim space
- Performance effect varies
- Problems with very large number of tables
- Less tested than default configuration



# innodb\_locks\_unsafe\_for\_binlog

- Can reduce lock contention dramatically
- Removes next-key locks
  - Phantoms possible for locking reads
- May break replication and recovery with binary log
  - Especially INSERT ... SELECT ...
- MySQL 5.1 should be safe to use with row level binary logging



# Speed up Shutdown

- InnoDB may take very long to shutdown
  - Flushing dirty buffers from buffer pool
- Increase downtime for upgrades etc
- **innodb\_max\_dirty\_pages\_pct**
  - Maximum percent of dirty pages
- **SET GLOBAL innodb\_max\_dirty\_pct=0**
  - Wait as dirty pages get close to 0, and shut it down



# SHOW INNODB STATUS

- The instrument for understanding what is going on inside InnoDB
- Partially exported as SHOW STATUS variables in MySQL 5.0
- Shows statistics about latches, locks, IO, logging activity, row level activity, thread queue activity etc.



# InnoDB and Hardware

- RAID With battery backed up cache may be important.
- NAS known to cause the problems
- May have problems scaling with many CPUs
  - Fix on a way
  - Faster CPUs, multiple low end boxes
  - Disabling HyperThreading may be good



# InnoDB Aware Schema

- Use short PRIMARY KEY
  - Long PK make all secondary index larger
- Have Primary key
  - InnoDB will use internal key anyway
  - And it will be 6 bytes in length
- Have sequential PRIMARY KEY
  - Non sequential inserts cause fragmentation



# Handling Long PRIMARY KEY

- If you have long primary key you can promote it to UNIQUE KEY
  - Add auto\_increment pseudo\_id column and make it primary key
  - Change real primary key to UNIQUE KEY
- **Note:** Lookups are slower by secondary key



# Power of PRIMARY KEY

- PRIMARY KEY is special key in InnoDB
- PRIMARY KEY lookups are much more efficient
  - Both in memory and IO bound
- PRIMARY KEY range scans have same speed as full table scans.
- Joins on PRIMARY KEYs are more efficient
  - Account in schema design



# No Key Compression

- InnoDB does not have key compression
  - As MyISAM does
- InnoDB Indexes can be 10 times larger than MyISAM indexes
  - One of the reasons InnoDB tables generally take 2-3 times more space
- Be easy on indexes
- May be fixed by gzip page compression



# Power of clustering

- Get benefit of clustering by primary key
- Messages Table
  - Primary key(user\_id,message\_id)
  - Very fast to get all messages for given user
  - Would be even better with multi column auto\_increment key support
- General rule: data which you need together to have close PK values



# Table Fragmentation

- InnoDB tables fragment over time
- Rows are not fragmented but pages can be scattered
  - Less of the problem because of large pages
- OPTIMIZE TABLE to rebuild the table
  - Slow (no index rebuilt by sort)
  - Blocks whole table during operation
  - Master-Master replication may help



# High Performance Backup

- Use physical level backup
  - Logical level backup is very slow to recover
  - But do NOT copy files with database running
- Innodb Hot Level Backup
  - Commercial solution
- LVM/Snapshot based backup
  - About same performance but free
  - Requires specific OS Setup



# Blob handling

- InnoDB can skip reading blobs if they are not in select column list
  - Makes sense to keep blobs in the same table
- Blobs stored each at separate page(s) if it does not fit to the page
  - Consuming at least 1 page
- Blobs are allocated in new space on update.



# Avoid count(\*) without where

- `SELECT COUNT(*) FROM TBL`
  - Instant for MyISAM, Memory etc
  - Slow for InnoDB
    - Performs table/index scan
- Try to avoid
  - `SHOW TABLE STATUS LIKE 'table'`
    - Approximate number of rows
  - Counter table for exact number of rows



# InnoDB Row count

- Count of rows is inaccurate
  - And guessed for each query execution
    - Using random BTREE dives
  - Can cause fluctuating plans
  - May be problem hard to catch.
- OPTIMIZE TABLE may help row count estimation accuracy



# InnoDB Statistics

- Cardinality values computed using BTREE dives as well
  - Can also be inaccurate
- Computed first time table is opened after start
  - Make first table open rather slow
- **ANALYZE TABLE** forces refresh
  - Using same estimation method



# InnoDB Performance Roadmap

- Fixing problems with multiple CPU Scalability
- Index rebuild by Sort
- Page Compression features
- Full Text Search
- Minor optimizations are constantly done



# Thanks for Coming

- Visit our Site
  - <http://www.mysqlperformanceblog.com>
- Send your follow up questions
  - [pz@mysqlperformanceblog.com](mailto:pz@mysqlperformanceblog.com)
- Consulting Requests
  - [consulting@mysqlperformanceblog.com](mailto:consulting@mysqlperformanceblog.com)

